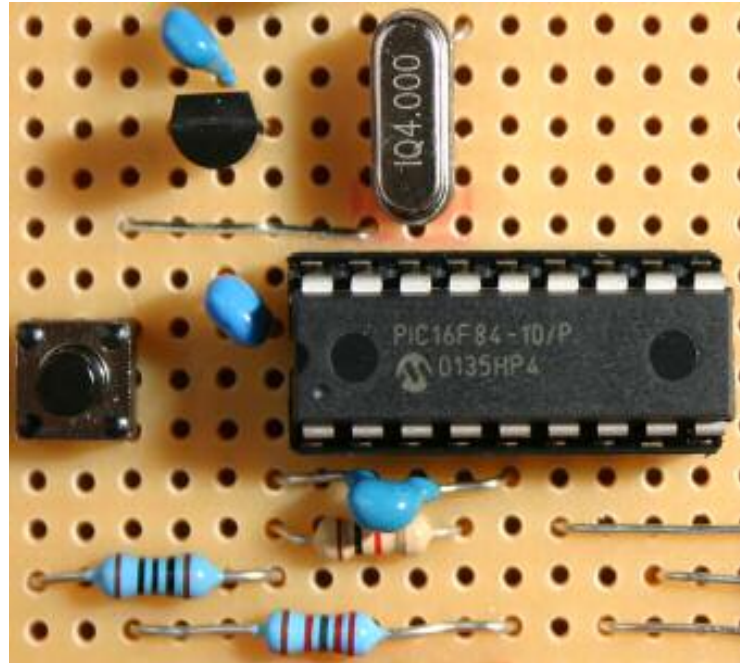


Development of a PIC Project.



By

Perry Andrews

January 2005

Table of Contents

Development of a PIC Project.....	1
Introduction.....	3
Analysis	4
Design.....	5
Development.....	9
Construction.....	12
Test	13

Table of Figures

Figure 1 - Development Process.....	3
Figure 2 - Turbo CAD Screen Shot	5
Figure 3 - CAD Symbols	5
Figure 4 - Dice Project Diagram.....	6
Figure 5 - Flow Diagram Symbols	6
Figure 6 - Smartdraw Diagram Package.....	7
Figure 7 - Dice Project Flow Diagram	8
Figure 8 - Matrix Multimedia PIC Development Board	9
Figure 9 - PIC Software Process.....	10
Figure 10 - Textpad 4 Screen Shot	10
Figure 11 - C2C PIC Development Environment.....	11
Figure 12 - Dice Project.....	12
Figure 13 - Final Constructed Dice Project	13

Introduction.

In this guide I will explain the process I use to develop and construct a PIC project. It is not the only way to work and there may be better ways. I hope to at least get you started and get you thinking on how to improve. The basic process after having an idea for a project is illustrated in the diagram below:



Figure 1 - Development Process

This is similar to the classic computer program development cycle with the exception that I specify separate develop and construct stages. In my development cycle I refer to writing the PIC program in the develop stage and build the project in the construct stage.

At the analyse stage you would write down from your original idea what you want the project to do. At the design stage you will take your analysis and design your project including the software and hardware. You will then develop the software and construct the hardware. Last but not least, you will test the project to make sure it works the way you intended in the analysis stage.

The process is iterative. That is it does not go from start to finish but goes back at times to revise a stage. For instance, during testing you may find a flaw in your design. You can go back and change the design and go through the process to the test stage again.

I like to use diagrams where I can to help simplify the development. I use the diagrams to break down the project into small pieces so they can be tackled easily. I use flow diagrams mainly even though they are a bit dated now. I will move on to using UML in later projects.

The following sections cover each stage in detail. I will use my Dice project as an example to show how the development is done.

Analysis

This is usually a more formal stage when developing a project for another person. If the project is for yourself you know pretty much what you want when you come up with the idea. When I have an idea I list down bullet points of what I want to achieve. This helps to make sure you don't forget something.

This was the analysis I did for the Dice project:

Traditional Dice.

- Display numbers on 7 led's representing dots on the dice.
- Press a button to generate a random number.
- While the button is pressed make the display 'roll' like a real dice rolls.

As you can see the list does not have to be exhaustive. A larger project like a burglar alarm will have many more 'features'.

Design

You now have a list of ‘features’ you want to have in your final project. What you need to do now is to turn your list into something you can make. The design has to include both the hardware and the software.

The hardware design is basically a circuit diagram. I use a CAD program called Turbo CAD to draw my circuit diagrams as it came free on the CD of a computer magazine. This is a bit over-kill but I am used to using the CAD software and it allows a nice diagram to be produced. You could use any drawing package but I would not use a paint type package. The software you use must allow you to easily change lines and move component symbols. The picture below shows Turbo CAD in use:

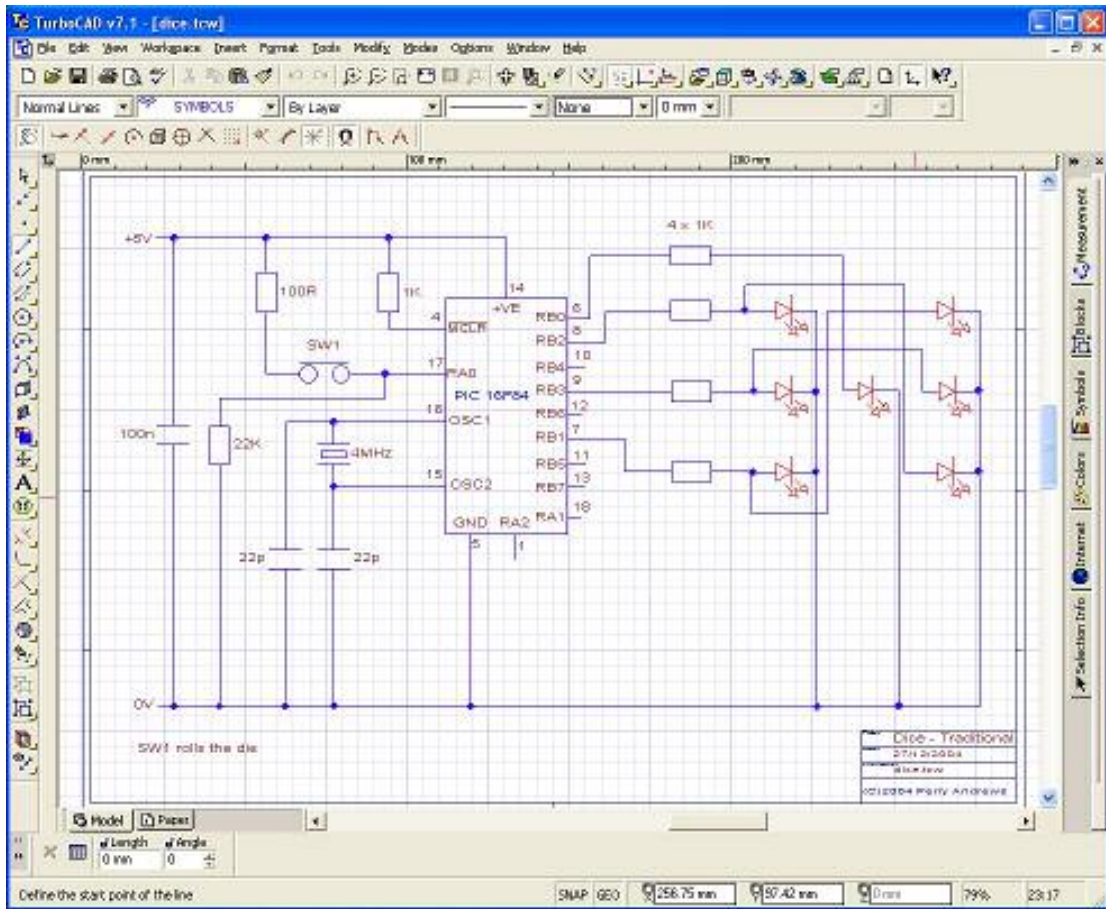


Figure 2 - Turbo CAD Screen Shot

I start by using a circuit diagram I did for one of my other projects. If this is your first project then you have to start from scratch. Every component must be drawn one by one. I use the following symbols to represent the components in my diagram:

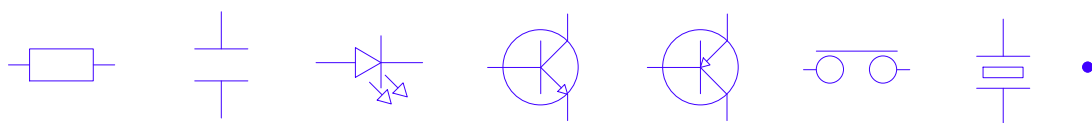


Figure 3 - CAD Symbols

This is not the full list of symbols. They are, from left to right, resistor, capacitor, led, npn transistor, pnp transistor, switch, crystal and connection. They should be easily recognised by anyone used to constructing electronic projects. The components are connected together using straight lines. These lines sometimes need to cross over. I try to avoid this by moving the components into different positions. Where the lines join I use a round dot to indicate there is a connection.

When I started the design for the Dice project I thought about using 7 outputs for the 7 led's. This is fine because I have enough outputs to do this. I thought more about it because I like to save on the inputs and outputs used. They are limited and I might want to use more of these later. I realised that some of the led's on the dice only ever light at the same time as others. The led in the middle can light on its own or with others so it needs its own output. The top right and bottom left led's always light together and are used when displaying 2, 3, 4, 5 & 6. Likewise, the top left and bottom right led's light together for the numbers 4, 5 & 6. Finally the middle left and right led's light together when displaying 6. So my dice now uses only 4 outputs instead of 7. The final diagram for the dice project is shown below:

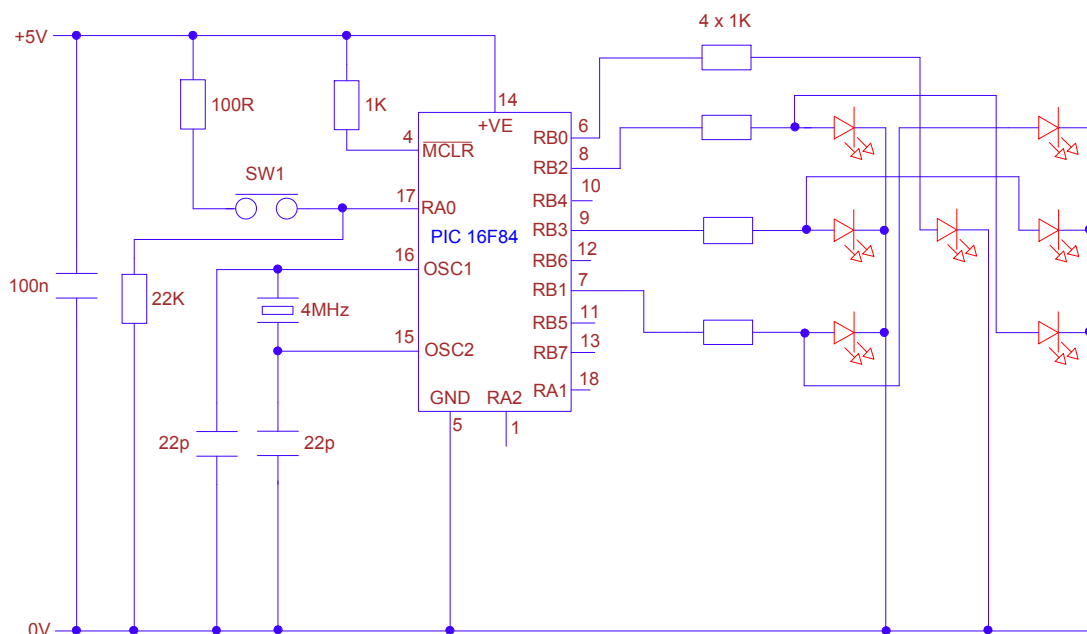


Figure 4 - Dice Project Diagram

Now I have a circuit diagram I can start to design the program. I do it this way because after the hardware design I know which outputs I need to program. If I find that some of my choices in the hardware design compromise my program or are just too complex to program, I can revise the hardware design.

I start the software design by creating a flow diagram. The diagram is created from the list of steps made in the first stage. I use the following flow diagram symbols:

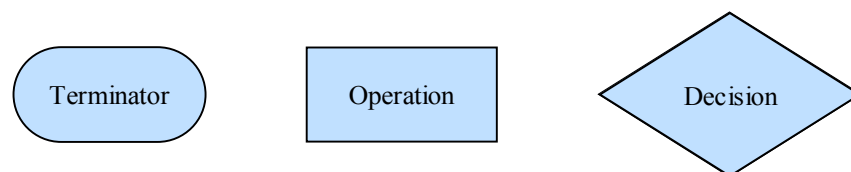


Figure 5 - Flow Diagram Symbols

Again a drawing package can be used to create the diagrams or they can be drawn with pencil and paper. I use a diagram software package called Smartdraw. This is tailored for diagrams and makes it easy to position and change the boxes. I got version 3 free on the cover disk of a magazine but I paid to upgrade to version 5. I find this easy to use and it comes with many symbols. It can even be used to create the circuit diagrams but I prefer the CAD software for this. Below is a screen shot of Smartdraw in operation:

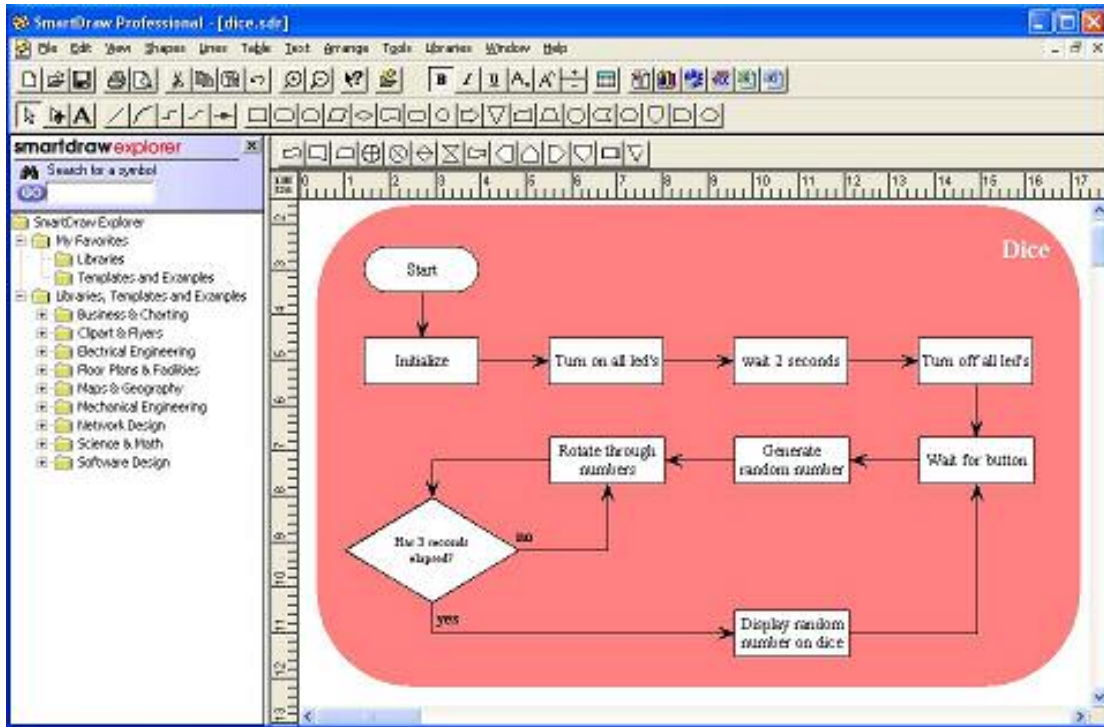


Figure 6 - Smartdraw Diagram Package

I start with the start terminator and move on to initialise the PIC microcontroller. I then use the symbols to plan the operation of the program. The flow moves from one operation to another. Sometimes the flow changes direction due to a decision. This is shown with the decision symbol which usually includes a question. If the answer to the question is yes the program flow moves to the operations indicated by yes otherwise the program flows down the no path.

There is usually no end terminator in my programs because they usually loop forever. The program ends only when the circuit is switched off.

I built up the diagram for the dice project by using the three points I made in the analysis stage. The idea is that a button is pressed to find and display a random number on the display. Therefore, the PIC must be initialised, the display cleared and the program must then wait for a button. When the button is pressed a random number is generated and the dice rolled. After a couple of rolls, the dice stops at the random number. The program then waits for the button to be pressed again. This is shown in the diagram below:

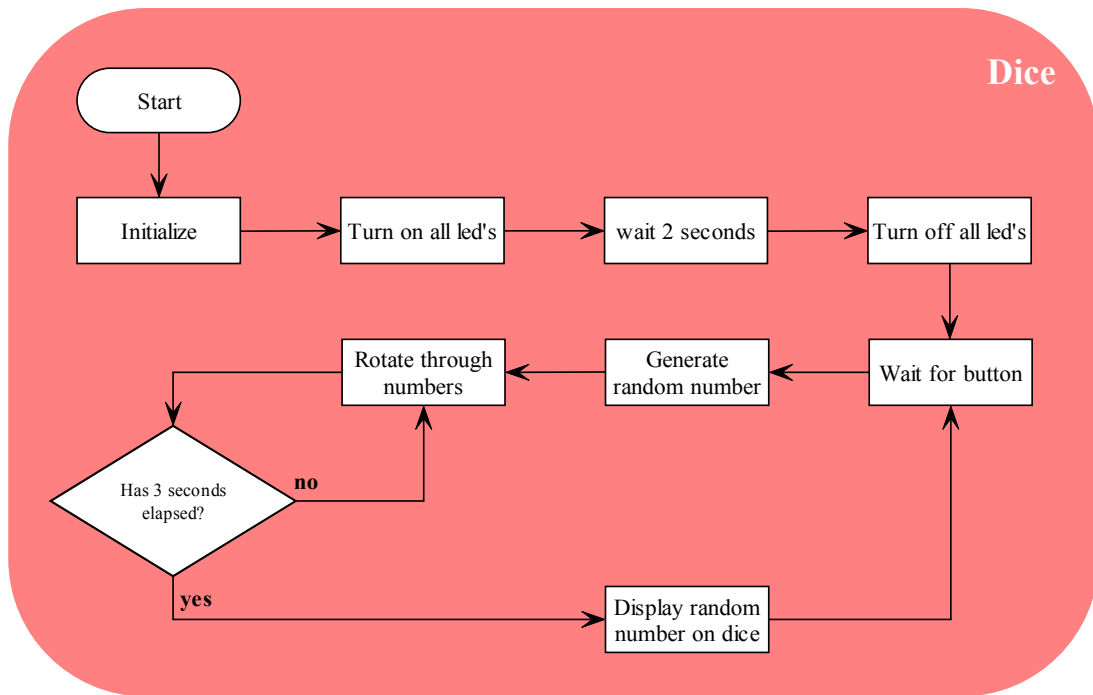


Figure 7 - Dice Project Flow Diagram

We now have the design for the hardware and the software and can now proceed to develop and construct the project.

Development

I create the program before constructing the project because I can change the design easier before construction if I find the program is difficult or impossible to write. I have not needed to do this yet but it could happen. I am able to do this because I use a dual purpose programmer and development board to develop the program. I download the program to the PIC using the programmer and then test using the development board. This means I do not have to remove the PIC micro until I am happy the program is working correctly. This will help protect the PIC micro as plugging in and out of the circuit will shorten its life.

The development board is made by Matrix Multimedia and can be used for a number of different PIC micros. I only use the 16F84 as it is now cheap and easy to use. The program can be easily erased and re-programmed. The programmer plugs into the Parallel port and software is supplied to program the PIC micro. A picture of the development board is shown below:

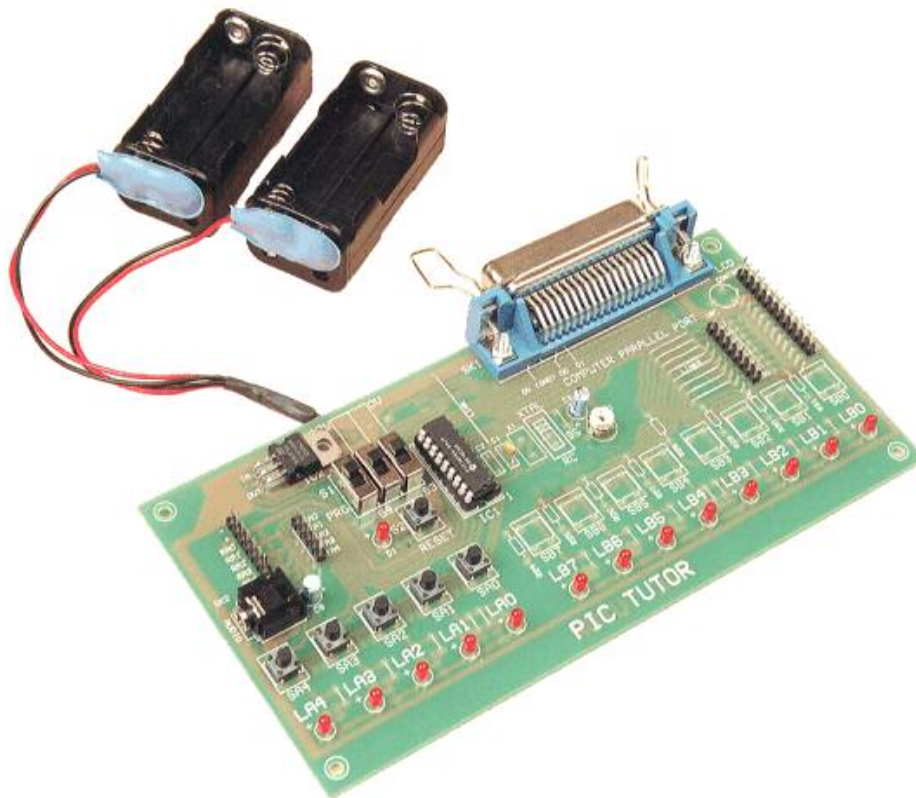


Figure 8 - Matrix Multimedia PIC Development Board

The program can be tested after downloading to the PIC micro by using the led's and switches on the board. This means I can test without having to construct the hardware first. The process to develop the program is shown in the diagram below:

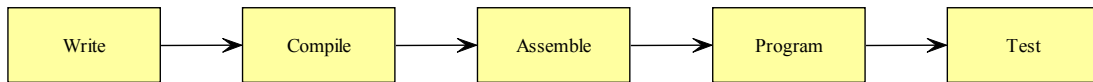


Figure 9 - PIC Software Process

I use a text editor called Textpad (shareware) to write the program because it has a number of features invaluable when writing programs. It highlights the language keywords etc to make the program easy to read. You can turn on line numbers which numbers each line. This makes debugging errors easier.

To write the program I use a template. This has a number of lines already included which are always required. These are the comment lines at the start, the main program procedure which must be used in every C program, and some lines to initialise the PIC. I then expand this program by writing a comment line for each of the operations in the flow diagram. If the operation appears more than once in a program I consider creating a separate procedure for this. I can then start writing the code to implement each of the operations and decisions.

If some of the program can be tested separately I will program the PIC and test before finishing the program. Again, by splitting the program into smaller pieces and test at the completion of each, the program will be easier to write. A screen shot of Textpad in action is shown below:

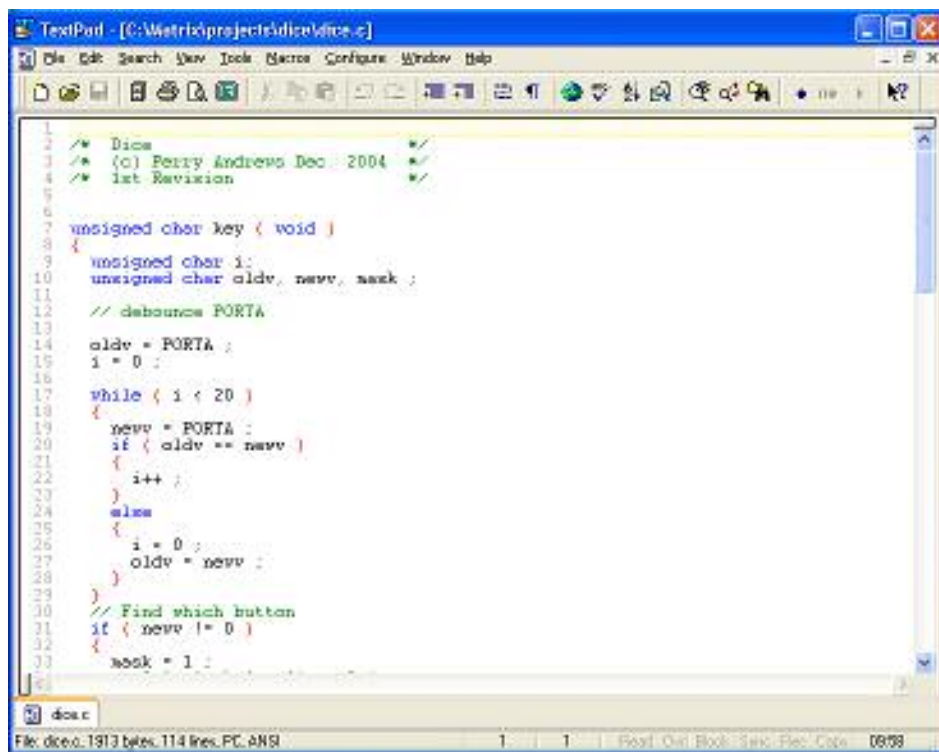


Figure 10 - Textpad 4 Screen Shot

The C program has to be converted into a form which the PIC can understand. This is known as compiling the program. I use C2C which is supplied with the development board. I open the C program in C2C and use the build option. The build option compiles the program into assembler, converts the assembler program into machine code and then sends the machine code to the PIC micro. These are all separate operations and separate programs but the C2C program can help do this in one step. These separate steps are described below but remember they are all performed within the C2C program.

Construction

I start by collecting together all the required components to make sure I have everything needed. These are then assembled on breadboard. This allows the components and circuit to be tested. It also gives an idea of component layout if the final project will be constructed on Vero Strip board. This is due to the similarity of layout between breadboard and strip board. The breadboard layout for the dice project is shown below:

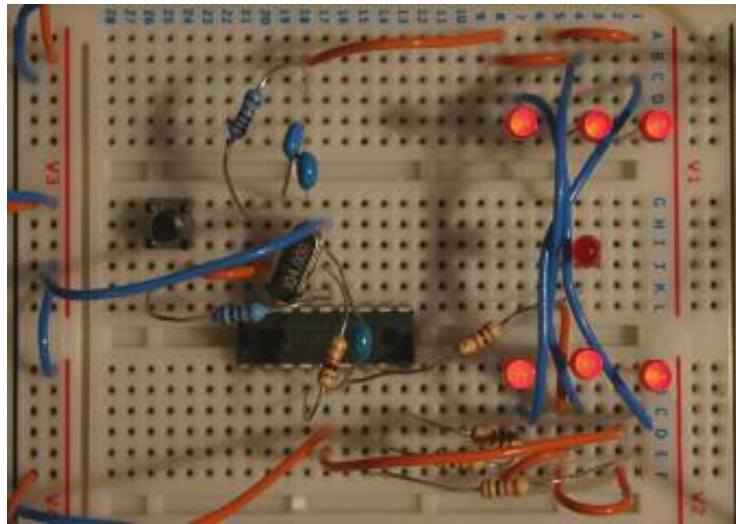


Figure 12 - Dice Project

Remember to keep the components leads at their original length so they can be used in the final project. This makes it tricky because you must keep the leads apart while testing otherwise they will short circuit. This could lead to the circuit working incorrectly or even damage to the components. So be careful!

I start with the PIC and work from each pin with the other components. I also put the pattern of leds in place before placing the resistors between the PIC and the leds.

It is not until the project is set up on breadboard do you really see what the final result will work like. The dice project for instance, has leds arranged in a pattern. This is the first time the pattern will be seen working. It might be the program will need a little tweaking at this point if the final components give a different result. Also, it might be the leds are too bright or dim so the resistors may need to be changed. It can be done easily at this stage when the components are just plugged in.

When the circuit works correctly, it can be transferred onto a more permanent installation. I use Vero Strip board as it is quick and easy to work with. I find it easy to transfer the components from the breadboard to the strip board. The dice project is shown in its final construction below:

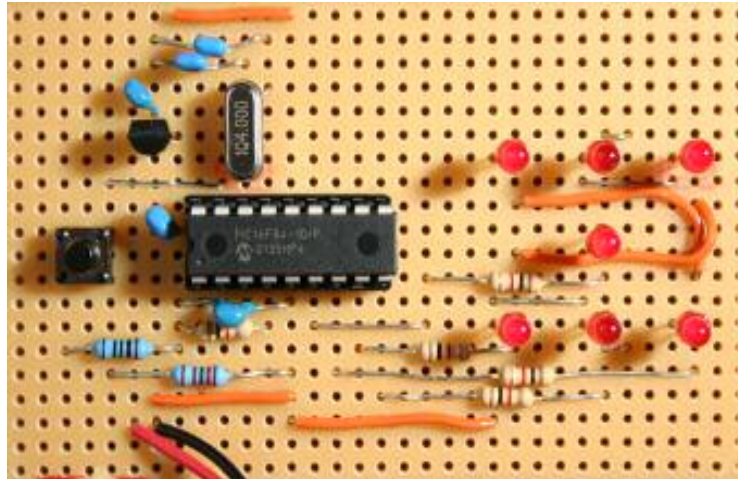


Figure 13 - Final Constructed Dice Project

The main difference between the strip board layout and the breadboard one is the components are laid out more neatly and close to the board. I like to keep the components in straight lines either across the strips or along them. I also measure the box and cut the strip board to size before starting construction.

In this case the first components to put in place are the LEDs. This is so they can be positioned neatly. You must remember to leave enough space for the other components. I then solder in the IC socket and the switch. Next are the resistors and capacitors. Finally the crystal and regulator are soldered in.

The strips have to be cut in places so that components can share a single track. For instance, the tracks need to be cut between the PIC pins.

When everything is finished, check for solder splashes. If you are sure everything is ok, plug in the PIC micro and apply power to test.

Test

In my projects the test stage occurs throughout the project. You test the program after programming the PIC micro and you test the circuit after assembling it on the breadboard. The final testing is to make sure the project does what you want it to do.